MODULE *DistributedLock*

EXTENDS *Naturals*, *FiniteSets*, *Sequences*, *TLC*

The set of clients

CONSTANT *Clients*

Client states

CONSTANTS *Active*, *Inactive*

Message types

CONSTANT *LockRequest*, *LockResponse*, *TryLockRequest*, *TryLockResponse*, *UnlockRequest*, *UnlockResponse*

An empty constant

CONSTANT *Nil*

The current lock holder

VARIABLE *lock*

The lock queue

VARIABLE *queue*

The current lock *ID*

VARIABLE *id*

$serverVars \triangleq \langle lock, id, queue \rangle$

Client states

VARIABLE *clients*

$clientVars \triangleq \langle clients \rangle$

Client messages

VARIABLE *messages*

Variable

VARIABLE *messageCount*

$messageVars \triangleq \langle messages, messageCount \rangle$

$vars \triangleq \langle serverVars, clientVars, messageVars \rangle$

$TypeInvariant \triangleq$

$\quad \land \forall c \in \text{DOMAIN } clients : Cardinality(clients[c].locks) \in 0 \,..\, 1$

```
$Pop(q) \triangleq SubSeq(q, 2, Len(q))$

$Send(m) \triangleq$
    $\land\ messages' = Append(messages, m)$
    $\land\ messageCount' = messageCount + 1$

$Accept(m) \triangleq$
    $\land\ messages' = Pop(messages)$
    $\land\ messageCount' = messageCount + 1$

$Reply(m) \triangleq$
    $\land\ messages' = Pop(messages) \circ \langle m \rangle$
    $\land\ messageCount' = messageCount + 1$
```

```
$HandleLockRequest(message) \triangleq$
    $\lor\ \land\ lock = Nil$
       $\land\ lock' = message$
       $\land\ id' = id + 1$
       $\land\ Reply([type \mapsto LockResponse, client \mapsto message.client, acquired \mapsto \text{TRUE}, id \mapsto id])$
       $\land\ \text{UNCHANGED}\ \langle queue, clientVars \rangle$
    $\lor\ \land\ lock \neq Nil$
       $\land\ queue' = Append(queue, message)$
       $\land\ Accept(message)$
       $\land\ \text{UNCHANGED}\ \langle lock, id, clientVars \rangle$

$HandleTryLockRequest(message) \triangleq$
    $\lor\ \land\ lock = Nil$
       $\land\ lock' = message$
       $\land\ id' = id + 1$
       $\land\ Reply([type \mapsto LockResponse, client \mapsto message.client, acquired \mapsto \text{TRUE}, id \mapsto id])$
       $\land\ \text{UNCHANGED}\ \langle queue, clientVars \rangle$
    $\lor\ \land\ lock \neq Nil$
       $\land\ Reply([type \mapsto LockResponse, client \mapsto message.client, acquired \mapsto \text{FALSE}])$
       $\land\ \text{UNCHANGED}\ \langle clientVars, serverVars \rangle$

$HandleUnlockRequest(message) \triangleq$
    $\lor\ \land\ lock = Nil$
       $\land\ Accept(message)$
       $\land\ \text{UNCHANGED}\ \langle clientVars, serverVars \rangle$
    $\lor\ \land\ lock \neq Nil$
       $\land\ lock.client = message.client$
       $\land\ lock.id = message.id$
       $\land\ \lor\ \land\ Len(queue) > 0$
             $\land\ \text{LET}\ m \triangleq Head(queue)$
               $\text{IN}$
```

```
                ∧ $lock' = m$
                ∧ $id' = id + 1$
                ∧ $queue' = Pop(messages)$
                ∧ $Reply([type \mapsto LockResponse,\ client \mapsto message.client,\ acquired \mapsto \text{TRUE},\ id \mapsto id])$
          ∨ ∧ $Len(queue) = 0$
            ∧ $lock' = Nil$
            ∧ $Accept(message)$
            ∧ UNCHANGED $\langle queue,\ id \rangle$
    ∧ UNCHANGED $\langle clientVars \rangle$
```

```
$IsActive(m) \triangleq clients[m.client] = Active$

$ExpireSession(c) \triangleq$
    ∧ IF $lock \neq Nil \land lock.client = c$ THEN
          LET $q \triangleq SelectSeq(queue,\ IsActive)$
          IN
              ∨ ∧ $Len(q) > 0$
                ∧ $lock' = Head(q)$
                ∧ $queue' = Pop(messages)$
              ∨ ∧ $Len(queue) = 0$
                ∧ $lock' = Nil$
                ∧ $queue' = \langle\rangle$
      ELSE
          ∧ $queue' = SelectSeq(queue,\ IsActive)$
          ∧ UNCHANGED $\langle lock \rangle$
    ∧ $clients' = [clients$ EXCEPT $![c].state = Inactive]$
```

```
$Lock(c) \triangleq$
    ∧ $clients[c].state = Active$
    ∧ $Send([type \mapsto LockRequest,\ client \mapsto c,\ id \mapsto clients[c].next])$
    ∧ $clients' = [clients$ EXCEPT $![c].next = clients[c].next + 1]$
    ∧ UNCHANGED $\langle serverVars \rangle$

$TryLock(c) \triangleq$
    ∧   $clients[c].state = Active$
    ∧   $Send([type \mapsto TryLockRequest,\ client \mapsto c,\ id \mapsto clients[c].next])$
    ∧   $clients' = [clients$ EXCEPT $![c].next = clients[c].next + 1]$
    ∧   UNCHANGED $\langle serverVars \rangle$

$Unlock(c) \triangleq$
    ∧ $clients[c].state = Active$
    ∧ $Cardinality(clients[c].locks) > 0$
    ∧ $Send([type \mapsto UnlockRequest,\ client \mapsto c,\ id \mapsto$ CHOOSE $l \in clients[c].locks :$ TRUE$])$
    $\land\ clients' = [clients\ \text{EXCEPT}\ ![c].locks = clients[c].locks \setminus \{\text{CHOOSE}\ l \in clients[c].locks : \text{TRUE}\}]$
    $\land\ \text{UNCHANGED}\ \langle serverVars \rangle$

$HandleLockResponse(message)\ \triangleq$
    $\land\ \lor\ \land\ message.acquired$
            $\land\ clients' = [clients\ \text{EXCEPT}\ ![message.client].locks = clients[message.client].locks \cup \{message.id\}]$
            $\land\ \text{UNCHANGED}\ \langle serverVars \rangle$
        $\lor\ \land\ \neg message.acquired$
            $\land\ \text{UNCHANGED}\ \langle clientVars,\ serverVars \rangle$
    $\land\ Accept(message)$

----

$Receive\ \triangleq$
    $\land\ Len(messages) > 0$
    $\land\ \text{LET}\ message\ \triangleq\ Head(messages)$
      $\text{IN}$
          $\lor\ \land\ message.type = LockRequest$
              $\land\ HandleLockRequest(message)$
          $\lor\ \land\ message.type = LockResponse$
              $\land\ HandleLockResponse(message)$
          $\lor\ \land\ message.type = TryLockRequest$
              $\land\ HandleTryLockRequest(message)$
          $\lor\ \land\ message.type = UnlockRequest$
              $\land\ HandleUnlockRequest(message)$

----

$Init\ \triangleq$
    $\land\ messages = \langle\rangle$
    $\land\ messageCount = 0$
    $\land\ lock = Nil$
    $\land\ queue = \langle\rangle$
    $\land\ id = 1$
    $\land\ clients = [c \in Clients \mapsto [state \mapsto Active,\ locks \mapsto \{\},\ next \mapsto 1]]$

$Next\ \triangleq$
    $\lor\ Receive$
    $\lor\ \exists\, c \in \text{DOMAIN}\ clients : Lock(c)$
    $\lor\ \exists\, c \in \text{DOMAIN}\ clients : TryLock(c)$
    $\lor\ \exists\, c \in \text{DOMAIN}\ clients : Unlock(c)$

$Spec\ \triangleq\ Init \land \Box[Next]_{vars}$

----
```

\ ∗ Modification History
\ ∗ Last modified *Fri Jan* 26 18:32:52 *PST* 2018 by *jordanhalterman*
\ ∗ Created *Fri Jan* 26 13:12:01 *PST* 2018 by *jordanhalterman*